A proposal learning strategy on CNN architectures for targets classification

Abdelmalek Toumi, Jean-Christophe Cexus, Ali Khenchaf

A proposal learning strategy on CNN architectures for targets classification

Abdelmalek Toumi, Jean-Christophe Cexus, Ali Khenchaf
ENSTA Bretagne - Lab-STICC, UMR CNRS 6285
2, Rue François Verny, 29806 Brest Cedex 9, France.
Email: {jean-christophe.cexus, abdelmalek.toumi, ali.khenchaf}@ensta-bretagne.fr

***Abstract*—Synthetic Aperture Radar (SAR) imagery has a great potential in remote sensing field in order to maintain aerial, land and maritime surveillance. To improve this potential for several goals, Machine Learning (ML) techniques are used for effective and efficient classification of remotely sensed imagery. The strengths of machine learning include the capacity to handle data of high dimensionality with several modality and to map classes with very complex and varied characteristics.**

We focus in this paper on the ship classification from SAR imagery using a deep learning algorithms with transfer leaning mechanisms. To improve the recognition performances, several learning strategies are proposed. We illustrate these issues through applying a transfer learning on deep learning ship classification with several learning strategies. We used in this work especially no huge labelled dataset available for training. The aim is thus to see how to leverage knowledge from models pre-trained on other tasks (source tasks) and use them for target classification (target task).

***Index Terms*—Automatic Target Recognition, Deep learning, SARShip Classification, Transfer learning, Learning strategy.**

I. Introduction

The environment characterization and fine description given by sensors for detection, localization, tracking targets (aircraft, ship, vehicle ...) and classification present an essential tasks for civil and military applications (on and/or under the sea surface). This characterization is thus important to improve aerial, land and maritime surveillance systems. Also, the threats generated by legal or illegal trade such as drug/weapons trafficking, illegal immigration, illegal fishing, pollution (hydrocarbons) or disaster management have shown the importance of a maritime/terrestrial surveillance areas. Thus, the areas surveillance constitutes today a very important stake for the states not only in the security sector, but also in the economic field. This surveillance can be carried out by means of airborne or satellite sensors, fixed and/or mobile, depending on the extent of the territory to be monitored and the targeted applications [1], [2].

In this study, we are interested in methods from Deep Learning and different architectures in a context of target classification on SAR images. In this context of the study, we focused on Machine Learning technologies in the context of Radar Remote Sensing problems related to the maritime domain. For detection, from SAR images, it is a matter of locating (targeting) the light points (if possible relevant) in large images (example of the reflectivity of ships more important than the sea surface). Classification is located further downstream from detection. It is in this step to discriminate (identify) more finely the detected objects. Classically, this step requires a higher level of feature extraction than the detection step. In this context, Convolutional Neural Networks (CNN) and transfer learning [3], [4] potentially offer a rather elegant and adequate solution to such a problem.

This paper is organized as follows. In section II we are also interested in the potential of transfer learning and learning strategies applied to proposed CNN architectures. In this section, we expose the proposed CNN architectures for target detection and classification/recognition on SAR images and their evaluation experimentation. In section IV, the obtained results and performances of the proposed DNN methods are presented and discussed. Finally, we give conclusions and perspectives.

II. Transfer learning Method for target Recognition

In this section, the Automatic Target Recognition system is introduced. We present briefly the principle of Deep Neural Network steps and describe the proposed Deep Neural Network approaches.The proposed CNN architecture which adopted from Zhang et al. [5] is presented by the figure 1 .

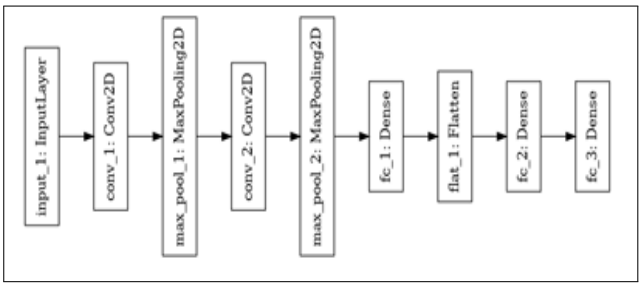

Fig. 1. "Zhang" Architecture [5].

This CNN architecture has the advantage of being simpler and well adapted to this work and evaluation of the different architectures presented in the following. Deep Neural Network which is specialized in object recognition and classification. During the feature extraction, the CNN architectures use the convolution operations to extract different features that are represented by generated filters. It is composed of 2 convolution layers, 2 Max Pooling layers and 3 fully connected layers and

uses ReLU type activations (see Figure 1 and the Table I for its descriptions).

TABLE I
CHARACTERISTICS OF THE "ZHANG" ARCHITECTURE.

Layer name	Channels	Kernel	Stride	Nb params
InputLayer	1	-	-	0
conv_1	32	3x3	1	288
max_pool_1	32	2x2	2	0
conv_2	64	3x3	1	18432
max_pool_2	64	2x2	2	0
fc_1	120	-	-	7680
flat_1	122880	-	-	0
fc_2	64	-	-	7896320
fc_3	n_class	-	-	64 * n_class

III. LEARNING STRATEGY

A. Early stopping techniques

The "early stopping" technique makes it possible to adjust the training process (number of epochs) according to the performance training evolution on the validation dataset. The objective is to train the network long enough to reach the best performance. To do this, we follow the evolution of the classification accuracy at each learning epoch, measured on the validation base. When it stops increasing, we stop the learning. We consider that the accuracy stops increasing when it does not increase during a predetermined number of epochs (Early Stopping number). Moreover, to ensure minimum learning, a minimum number of learning epochs is defined. When the training process is stopped, the training parameters which give us the best performance (accuracy) with the "best epoch" in the training process are retained and stored.

B. Setting the learning rate

To adjust the learning rate in order to get the most out of each learning, the "1-cycle" method is used. This strategy method is introduced by [6] and then recommended by Fastai [7]. To study this method and compare it with more classical methods of learning rate adjustment, several simulations are performed. We note that, in the "1-cycle" method, the learning rate is varied and changed during the learning process, over a cycle: first by increasing the learning rate and then by decreasing it. This method is thus parameterized by different hyperparameters below:

- The maximum value of the learning rate (the one reached in the middle of the cycle).
- The starting ratio: determines the initial value of the learning rate in relation to the maximum value of learning.
- The final ratio: determines the final value of the learning rate in relation to the maximum value of learning rate.
- The length of the cycle.
- The share of the rising part of the cycle in relation to the total.
- The function used to increase and decrease the learning rate.

In this work, we have used the cosine function as a function for the increase and decrease of the learning rate in the learning cycle [7]. To illustrate this method, the Figure 2 shows the evolution of the learning rate following the "1-cycle" method with the following hyperparameters:

- Function used : cosine function.
- Maximum value of the learning rate : 0.003.
- Starting ratio : $1/25$.
- Final ratio: $1/300$.
- Cycle time: 100 epochs.
- Share of the rising part of the cycle compared to the total: 25%.

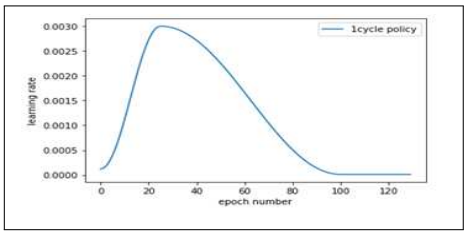

Fig. 2. The evolution example of the learning rate in the learning process for "1-cycle" method.

To evaluate the 1-Cycle strategy method, some experimentation's are made and presented in the next section.

C. Classification Performance Measurement

In this paper, the performance measurements are evaluated for the proposed method on ship classification from SAR images using the OpenSARShip dataset. This dataset is limited in classes number and class representation (images/class) [8]. The small dataset and a reduced number of images per class leads to problems and poor performance measurements. Indeed, we generally obtain a wide variety of results for a given training setup on several independent achievements. This is due to the randomness of the setup initialization (such as the initialization of the network parameters) and to the learning optimization. In the other hande, the small number of images in the test dataset can induce a high variability of the network performances. Indeed, if only some images are well recognized (or false recognized) can increase (or decrease) significantly the performance measurements.

To address this problem, we therefore evaluate performance measurements by reproducing the learning process and performance measurement several times for a given learning configuration. These realizations are done independently. In order to obtain a number of samples that allows to correctly represent the performance, we choose to perform 15 independent training realizations for each training configuration using TensorFlow v2.3.0 Library.

To illustrate this proposal, we present in Table II a set of performance measurements obtained from a given training configuration which the significant variability on results can be observed. So, we present in Table II the 15 realizations and their classification accuracy (noted $\texttt{Acc}$). This accuracy distribution is presented by Figure 3 which we visualize the estimation of the associated probability density (using

a Gaussian kernel) as well as the average values to make comparison.

TABLE II
CHARACTERISTICS OF THE "ZHANG" ARCHITECTURE.

Realization number	1	2	3	4	5	6
Acc %	64.7	92.3	96.2	95.1	95.9	56.9
Realization number	7	8	9	10	11	12
Acc %	94.6	96.0	86.6	86.4	96.0	64.9
Realization number	13	14	15	-	-	-
Acc %	90.6	95.5	81.2	-	-	-

Once the samples have been obtained, it was necessary to determine a simple way to represent the corresponding distribution. We did not choose the mean / standard deviation pair. Indeed, the mean is sensitive to the value of the extreme cases and can therefore vary a lot for two measurements series of the same training configuration. We thus used the median of the distribution, which does not have this disadvantage.
Moreover, to represent the samples variability linked to the median, given the asymmetry of the distributions encountered in this study, we have used a asymmetrical difference averages: for each sample, we calculate the value $(sample - median)$ and then estimate the average of the obtained negative values as well as the average of the obtained positive values. We thus obtain the average difference of the samples to the median below the median on the one hand and for the samples above the median on the other hand. The Figure 3 illustrate an example to illustrate the performance measurement problem.

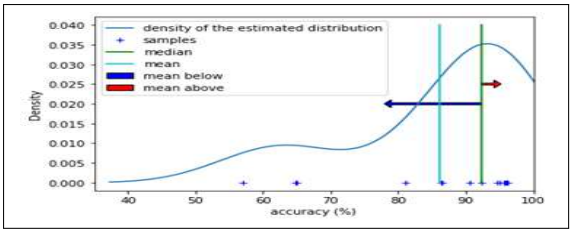

Fig. 3. Probability density estimation of the performance measurement distribution.

As summary of this step, we estimate the training performances on 15 times (independently) for a given configuration. Then, we obtain 3 quantitative indicators that represents the corresponding measurements:

- Median.
- Average of differences $(sample - median)$ that are negative, the "negative mean difference".
- Average of differences $(sample - median)$ that are positive, the "positive mean difference".

To complete the evaluation performances of the proposed model, we present in the next section the effect of the activation function.

D. The Activation Function Modification

We replace the ReLU activation function of fc_2by a Leaky-ReLU function in order to "force" the different channels of fc_2 to be non-zero in order ton increase the classification accuracy [9]. The $ReLU$ and $Leaky - ReLU$ activation functions are represented on the Figure 4.

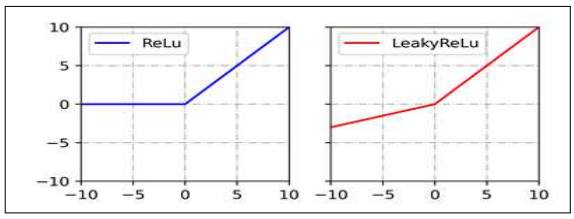

Fig. 4. ReLU and Leaky-ReLU activation functions.

IV. RESULTS AND DISCUSSIONS

In this section, we present the different simulations obtained in order to evaluate the strategies methods presented in the previous section. So, we discuss the different results obtained in order to dress best practice rules.

A. Experimental Setup

For the use of transfer learning in the context of SAR images, the MSTAR dataset [10] is used. The MSTAR database (Moving and stationary Target Acquisition and Recognition) brings together a set of images collected in 1996 in X band $(8 - 12\text{GHz})$ with an HH polarization and a resolution of 30 cm. The MSTAR base consists of 5165 images of 10 military vehicles (classes). The size of images is 128×128 pixels. The train dataset contains 2740 images (53%), and the test dataset contains 2425 images (47%). As second dataset, the OpenSARShip (version 1.0) set [8] is used in order to perform ship classification due to its public availability and thus the potential for performance comparison with other works. This dataset can then also be used as an intermediate dataset for further maritime datasets, in order to perform transitive transfer learning; providing a more similar task to other ship classification problems than MSTAR. The OpenSARShip database is a medium-resolution ship dataset that contains 11346 image ships cropped from a total of 41 Sentinel-1 images, and it covers mainly 5 ports in Asia and it has 17 ships classes (AIS types). The image sizes range from 9×9 pixels to 445×445 pixels, with 80% of the image sizes ranging from 33×33 to 169×169 pixels.But, the class distribution of the whole OpenSARShip database are not uniformly distributed [8]. Therefore, we have select 4 classes ships from the initial classes which are : Cargo (with 925 images), Container Ship (with 218 images) and Bulk Carrier (with 813 images).

B. Comparison of strategies for learning rate

To choose the "1-cycle" method presented in the section III, we performed simulations to compare different learning rate tuning methods: Constant, Decreasing and "1-cycle". The corresponding evolution of the learning rate during training is illustrated by the Figure 2. The 3 learning rate setting scenarios are configured as follows:

- Constant learning rate:
  - Learning rate : 3*10-3.
- Decreasing learning rate:
  - Maximum value of the learning rate : 3*10-3.

  - Learning rate decrease period: 1 epoch.
  - Learning rate decrease factor: 0.8.
- Learning rate ” 1-cycle:
  - Maximum value of the learning rate: 3*10-3.
  - Starting factor : 1/25.
  - Final factor: 1/300.
  - Cycle time: 20 epochs.
  - Share of the rising part of the cycle compared to the total: 25%.
  - Function used to increase then decrease the learning rate : cosine.

To perform this comparison, we have used the ”Zhang” architecture (CF. Figure 1) on MSTAR dataset to perform a simulations process. Then, we measure the performance of the trained network as well as the number of epochs required to trigger the learning interruption (by the early stopping mechanism). The learning configuration of the architecture remains identical (except for the learning rate):

- Batch size: 32.
- Optimizer: Adam.
- Minimum number of epochs of learning (early stopping): 20.
- Minimum number of epochs without improvement to declare learning complete (early stopping): 15.

The results of the corresponding simulations are given by Table III. The median accuracy obtained is higher for the ”1-cycle” learning rate than in the other cases. We can also see that the learning performed with this method presents results with a lower variability compared to the other methods. Finally, we notice that the learning converges more quickly with this method. Note that these elements support the results of [7] which recommends, as a good practice, to use the ”1-cycle” learning rate adjustment method.

TABLE III
RESULTS FOR THE COMPARISON OF LEARNING RATE ADJUSTMENT METHODS.

	n_epoch	accuracy
Constant learning rate		
median	15.0	92.8%
mean_below	-5.9	-6.9%
mean_above	15.0	2.4%
Decreasing learning rate		
median	14.0	92.3%
mean_below	-3.0	-14.4%
mean_above	9.2	2.9%
Learning rate " 1cycle		
median	10.0	94.9%
mean_below	-1.4	-1.5%
mean_above	1.1	1.0%

C. Comparison of strategies for Transfer Learning

We measure the performance of the ”Zhang” architecture in classification only on the OpenSARShip dataset due to lack of space. However, we note that the learning step on the ”Zhang” architecture provides a good classification accuracy on the MSTAR dataset with a median accuracy around 94.8% on the test dataset. Six learning strategies are performed: without and with a learning transfer from MSTAR to OpenSARShip:

- The model is fully trained with a random initialization of the network parameters. We noted this model ”from scratch”.
- The model is initialized with the trained parameters from the model trained from the MSTAR dataset. Then, all the layers of this model are re-trained. We noted this model ”no freeze”.
- The model parameters are initialized from the training on the MSTAR dataset. Then all layers of the model are re-trained except the layers between the input and the conv_1 layer (included). We noted this model ”freeze conv_1”.
- The model parameters are initialized from the training on the MSTAR dataset. Then all layers of the model are re-trained except the layers between the input and the conv_2 layer (included). We noted this model ”freeze conv_2”.
- The model parameters are initialized from the training on the MSTAR dataset. Then all layers of the model are re-trained except the layers between the input and the fc_1 layer (included). We noted this model ”freeze fc_1”.
- The model parameters are initialized from the training on the MSTAR dataset. Then all layers of the model are re-trained except the layers between the input and the fc_2 layer (included). We noted this model ”freeze fc_2”.

We used the following hyperparameters for these learnings:

- Batch size: 32.
- Optimizer: Adam.
- Learning rate ”1-cycle”:
  - Maximum value : 0.003.
  - Cycle time: 20 epochs.
  - Share of the upstream phase: 25%.
  - Initial ratio: 1/25.
  - Final ratio: 1/300.
  - Rise and fall function: cosine.
- Early stopping:
  - Minimum number of epochs: 20.
  - Number of epochs without improvement to stop learning: 15.

The results for OpenSARShip are given on the Table IV. It can be observed that the learning is not as good as on the MSTAR database on the OpenSARShip database (from scratch scenario), which is explained by the small amount of annotated data available.
The transfer learning compensates for this effect, with a slight improvement in classification accuracy in the ”no freeze”, ”freeze conv_1” and ”freeze conv_2” cases. The transfer scenarios ”freeze fc_1” and ”freeze fc_2” do not improve the classification performance. In particular, we notice that the performance is very degraded for ”freeze fc_2”. This is also visible on the corresponding confusion matrix: the objects are mostly misclassified and no object is classified in the ”Cargo”

class. We also notice from the observation of the confusion matrices that the ”no freeze”, ”freeze conv_1” and ”freeze conv_2” cases improve the performance over ”from scratch” by improving the performance on ”Cargo” and ”Container Ship”, the minority classes.

TABLE IV
PERFORMANCE OF THE ”ZHANG” ARCHITECTURE ON THE OPENSARSHIP DATASET FOR THE DIFFERENT TRANSFER LEARNING SCENARIOS.

OpenSARShip	from scratch	no freeze	freeze conv_1	freeze conv_2	freeze fc_1	freeze fc_2
median	72.8%	**75.7%**	**74.6%**	73.4%	71.6%	41.4%
mean_below	-0.8%	-2.6%	-2.8%	-2.1%	-2.1%	-3.3%
mean_above	1.2%	0.4%	1.1%	2.0%	2.4%	6.5%

If the global performance is not improved when the number of frozen layers is increased, we notice however a change in the distribution of the classes which are correctly classified. The performance is indeed improved for the minority classes when we increase the number of frozen layers at the cost of a degraded performance for the majority class; this results in a degraded global performance. To go further, we could explore the compensation mechanism of the unequal distribution of the classes in the OpenSARShip database.

D. Comparison of strategies for activation function

In this simulation, the activation function is replaced and the architecture is trained with a random parameter initialization. The results are presented in the Table V and VI.

TABLE V
THE PERFORMANCES COMPARISON ON MSTAR DATASET ON ORIGINAL ”ZHANG” MODEL USING THE LEAKY-RELU ACTIVATION FUNCTION IN THE FC_2 LAYER WITH A DROPOUT (30, 50, 80).

	Original	LeakyReLu	LeakyReLu_Dropout30	LeakyReLu_Dropout50	LeakyReLu_Dropout80
median	94.8%	**95.1%**	94.8%	94.4%	**95.8%**
mean_below	-1.6%	-0.7%	-0.2%	-0.4%	-0.5%
mean_above	0.6%	0.3%	0.5%	0.6%	0.6%

TABLE VI
THE PERFORMANCES COMPARISON ON MSTAR DATASET USING DROPOUT ALONE WITH THE COUPLED USE OF DROPOUT AND THE LEAKY-RELU ACTIVATION FUNCTION IN THE FC_2 LAYER.

	orig_Dropout30	LeakyReLu_Dropout30	orig_Dropout50	LeakyReLu_Dropout50	orig_Dropout80	LeakyReLu_Dropout80
median	92.4%	**94.8%**	86.6%	**94.4%**	20.7%	**95.8%**
mean_below	-2.3%	-0.2%	-22.1%	-0.4%	-3.1%	-0.5%
mean_above	0.4%	0.5%	4.0%	0.6%	3.8%	0.6%

We observe a slight improvement in the results coupled with a decrease in the variability of the results. The fc_2 channels with Leaky-ReLU are indeed non-zero but still appear underutilized given the small performance gain obtained. So, to take this further, we couple this change in activation function with the use of Dropout right after fc_2. Dropout allows to randomly remove a configured proportion of its input activations during learning. The idea is thus to ”force” the learning to take advantage of the activations of fc_2, in addition to having ”forced” them to be non-zero with the Leaky-ReLU activation function.
We tried several proportions of Dropout for our simulations: 30%, 50% and 80%. This proportion corresponds to the share of activations that are dropped (and not those kept). The corresponding results are presented in the Table V. We observe in the Dropout 80% case a clearer improvement in performance obtained. To make sure that this improvement results from the coupling of the Leaky-ReLU function in the fc_2 layer with the Dropout 80% and not from the Dropout 80% alone, we have also performed the corresponding simulations with the Dropout 80% alone. The corresponding results are presented in the Table VI. So we observe that it is the coupling of the Leaky-ReLU activation function in the fc_2 layer with the 80% Dropout that leads to the performance improvement and not the addition of the 80% Dropout alone.

E. Comparison of classification Architectures

We have compared different architectures frequently used in the optical image domain for the case of our problem: VGG16 [11], ResNet50 [12], Xception [13], DenseNet121 [14], EfficientNetB0 [15] and MobileNetV2 [16]. These architectures are often much more complex and therefore require a substantial learning curve in order to exploit their potential. We therefore use these architectures with pre-trained networks on ImageNet whose pre-trained parameters are available from the Keras library. We compare these architectures in two different learning configurations: Fixed CNN and non-fixed CNN. The frozen CNN configuration corresponds to the case where the convolutional part of the architecture is frozen (its parameters are not modified by the learning) and the ”softmax” classifier part is adapted to the target task. The non-fixed CNN configuration corresponds to the case where all the parameters of the network are adapted to the target task. In order to measure the performance in these different scenarios, we use the performance measurement method explained above. Learning is performed using the early stopping and learning rate adjustment methods explained above. We used the following hyperparameters:

- Batch size: 32.
- Optimizer: Adam.
- Learning rate ”1-cycle”:
  - Maximum value : 0.001.
  - Cycle : 40 epochs.
  - Share of the upstream phase: 25%.
  - Initial ratio: 1/25.
  - Final ratio: 1/10.
  - Rise and fall function: cosine.
- Early stopping:
  - Minimum number of epochs: 40.
  - epochs Number without improvement to stop learning: 10 epochs.

We review the obtained results from these architectures for the two data sets: MSTAR and OpenSARShip. The results for the MSTAR database are given by the Table VII (frozen CNN) and the Table VIII (not frozen CNN). The results for the OpenSARShip database are given by the Tableau IX (frozen CNN) and the X (unfixed CNN).
It can be noted that the results strongly depend on how the pre-trained parameters are processed. Thus, for MSTAR,

all architectures obtain better performances in the unfrozen CNN case except MobileNetV2. On the other hand, we do not observe this phenomenon for OpenSARShip. It can also be noted that the ResNet50 and DenseNet121 architectures achieve the best performance for MSTAR (in the non-fixed CNN case) while for OpenSARShip it is MobileNetV2 that achieves the best performance (in the fixed CNN case). For these two data sets, the performances obtained are higher or equal to the performances obtained with the "Zhang" architecture. This is due to the complexity of the architectures evaluated here, which have a much higher potential to extract complex features from images and thus a higher potential for best classification performance.

TABLE VII
COMPARISON OF "CLASSICAL" ARCHITECTURES ON THE MSTAR DATASET WITH TRANSFER LEARNING FROM IMAGENET (FROZEN CNN).

	VGG16	ResNet50	Xception	DenseNet121	EfficientNetB0	MobileNetV2
median	64.0%	40.8%	70.6%	79.3%	16.9%	87.0%
mean_below	-0.6%	-1.9%	-0.3%	-0.4%	-1.4%	-0.2%
mean_above	0.6%	1.2%	0.2%	0.5%	1.4%	0.2%

TABLE VIII
COMPARISON OF "CLASSICAL" ARCHITECTURES ON THE MSTAR DATASET WITH TRANSFER LEARNING FROM IMAGENET (NOT FROZEN CNN).

	VGG16	ResNet50	Xception	DenseNet121	EfficientNetB0	MobileNetV2
median	93.2%	**98.1%**	**97.6%**	**98.3%**	96.6%	63.3%
mean_below	-4.3%	-0.4%	-1.7%	-0.8%	-1.6%	-13.7%
mean_above	2.5%	0.3%	0.4%	0.2%	1.1%	17.5%

TABLE IX
COMPARISON OF "CLASSICAL" ARCHITECTURES ON THE OPENSARSHIP DATASET WITH TRANSFER LEARNING FROM IMAGENET (FROZEN CNN).

	VGG16	ResNet50	Xception	DenseNet121	EfficientNetB0	MobileNetV2
median	68.6%	74.6%	66.9%	72.8%	71.0%	76.3%
mean_below	-2.0%	-1.1%	-0.9%	-3.0%	-1.1%	-0.7%
mean_above	0.9%	0.6%	0.7%	0.7%	0.3%	1.1%

TABLE X
COMPARISON OF "CLASSICAL" ARCHITECTURES ON THE OPENSARSHIP DATASET WITH TRANSFER LEARNING FROM IMAGENET (NOT FROZEN CNN).

	VGG16	ResNet50	Xception	DenseNet121	EfficientNetB0	MobileNetV2
median	75.7%	69.8%	73.4%	72.8%	68.0%	64.5%
mean_below	-0.8%	-1.0%	-2.0%	-3.0%	-4.3%	-2.2%
mean_above	0.9%	2.4%	1.8%	2.1%	1.9%	1.6%

V. CONCLUSIONS

The work presented in this paper shows the effectiveness of these Deep learning methods for the target recognition. The recent great success of Deep Convolutional Neural Network in several application fields, improves their efficiencies to extract a feature descriptors by constructing a hierarchical network to automatically learn hierarchical features from MSTAR and OpenSARShip datasets. This work highlights the use of several learning strategies are proposed to perform the performances in SAR image classification. This work highlights the trade-off between complexity and learning capacity to be achieved. This trade-off can be influenced by different techniques to counteract the overlearning phenomenon, such as unsupervised learning, self-supervised learning, data augmentation, and transfer learning.

In the future work, the optimization and improvement of the methodology developed in this study will be extended and completed. The simulation results demonstrate the relevance and robust nature of the approach. However, for our future work, the performances and results of proposed approach on other cluttered images will be experimented with a combined CNN architectures.

ACKNOWLEDGMENT

The authors would like to thank the AID-DGA (Agence de l'innovation de défense, Direction Générale de l'Armement France) for the support to this research.

REFERENCES

[1] S. Zaied, A. Toumi, and A. Khenchaf, "Target classification using convolutional deep learning and auto-encoder models," in *4th International Conference on Advanced Technologies for Signal and Image Processing*, 2018, pp. 1–6.

[2] J.-C. Cexus, A. Toumi, and M. Riahi, "Target recognition from ISAR image using polar mapping and shape matrix," in *5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2020, pp. 1–6.

[3] Z. Huang, Z. Pan, and B. Lei, "Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data," *Remote Sensing*, vol. 9, no. 9, p. 907, 2017.

[4] Z. Huang, C. O. Dumitru, Z. Pan, B. Lei, and M. Datcu, "Classification of large-scale high-resolution SAR images with deep transfer learning," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 1, pp. 107–111, 2020.

[5] D. Zhang, J. Liu, W. Heng, K. Ren, and J. Song, "Transfer learning with convolutional neural networks for SAR ship recognition," in *IOP Conference Series: Materials Science and Engineering*, vol. 322, no. 7. IOP Publishing, 2018, p. 072001.

[6] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[7] J. Howard and S. Gugger, "Fastai: a layered api for deep learning," *Information*, vol. 11, no. 2, p. 108, 2020.

[8] L. Huang, B. Liu, B. Li, W. Guo, W. Yu, Z. Zhang, and W. Yu, "Opensarship: A dataset dedicated to sentinel-1 ship interpretation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 1, pp. 195–208, 2017.

[9] A. K. Dubey and V. Jain, "Comparative study of convolution neural network's relu and leaky-relu activation functions," in *Applications of Computing, Automation and Wireless Systems in Electrical Engineering*. Springer, 2019, pp. 873–880.

[10] E. R. Keydel, S. W. Lee, and J. T. Moore, "MSTAR extended operating conditions: A tutorial," in *Algorithms for Synthetic Aperture Radar Imagery III*, vol. 2757. International Society for Optics and Photonics, 1996, pp. 228–242.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[14] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 4700–4708.

[15] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.